Smarter Anti-Entropy with Bandits

Benjamin Bengfort and Pete Keleher

Abstract

We propose an *adaptive* distributed storage system that can monitor user access patterns and network environment and change its replication behavior in order to improve consistency. Specifically we look at eventually consistent replication implemented with routine, pairwise anti-entropy sessions. By applying reinforcement learning techniques to the anti-entropy process, we show that we can optimize the speed of replication, thereby reducing overall inconsistency. Additionally we explore monitoring mechanisms that modify the reinforcement learning process, detecting anomalies to temporarily shift behavior or to reset reinforcement learning. We show the advantages of such a system through an implementation called Honu, a replicated key-value store.

Introduction

The next phase of distributed storage systems will be smarter than their predecessors by optimizing their behavior in response to their environment. Constraints such as bandwidth, latency, storage, mobility, and user access patterns mean that a future data fabric of heterogenous devices and systems can shape how they participate in full or partial replication. User experience, consistency, fault tolerance, and availability can all be improved by *adaptivity*.

An adaptive system monitors its environment and changes its behavior at runtime without user intervention. For example, a system colocated with a mobile user may detect that it is offline and track changes in objects being modified. When the system comes back online, it uses merging algorithms during replication to ensure data integrity. This example, however, is a simple heuristic mechanism. Truly adaptive systems will use machine learning techniques both to identify criteria for response as well as to learn correct behaviors. We believe a powerful combination for such a system is anomaly detection algorithms [2, 11] for monitoring combined with online reinforcement learning [9, 7, 4, 8] to suggest behavioral strategies.

In this paper we consider the adaptive behavior of an eventually consistent object store implemented with pairwise anti-entropy [5, 12]. In such a system, clients access objects on one or more replicas in the system, observing the current local state or making a local update. The state is replicated on a routine basis, where one replica randomly selects another and exchanges state to ensure each has the latest version. The system is eventually consistent [13] because so long as there are no accesses, the system will reach a consistent state. Anti-entropy is a specific type of gossip protocol [6] that is deterministic and avoids broadcast saturation problems. However, during runtime, the system is prone to *stale reads* and *write forks* that are observable to external clients [3].

The speed of total replication throughout the system determines the severity of such inconsistencies. Said another way, the faster an update can replicate through the system, the fewer inconsistencies external clients will encounter. However, because anti-entropy is a random process, no guarantees can be made about how quickly an update propagates.

Instead of using uniform random selection, however, we propose to use *bandit algorithms* to modify the selection of peers to perform replication. These algorithms will learn optimal peers to have anti-entropy sessions with by optimizing rewards based on availability, amount of data exchanged, latency of connection and locality of access. We will show that the topology of messaging in an anti-entropy session will change over time to respond to client accesses as optimally as possible, minimizing the total replication delay.

The location of accesses in such a system will also change over time. Therefore an optimal configuration reached through reinforcement learning may not always be the optimal configuration. We therefore propose to use anomaly detection algorithms, particularly *one-class SVMs* to detect changes in user behavior. The SVM state will consider the location and frequency of accesses in a system. If the access pattern changes, the system will revert it's peer selection probabilities, learning a new optimal configuration.

System Description

Our system will consist of N peers that maintain a set of replicated objects. Clients can contact one or more replicas to `Get` or `Put` data to/from an object. On a `Get` operation, the replica(s) contacted will return the *latest* version of the object. On a `Put` operation, the replica(s) contacted will update a monotonically increasing conflict-free distributed version number [1, 10] which determines which version is later than another version. Objects also contain parent versions, which delineates their lineage in a read-then-write context. the parent version is the latest version of the object on the replica.

On a routine interval, T, each replica will select a random peer and perform *bilateral anti-entropy*. That is, the replica will send a vector of all latest versions of objects to the remote peer. The remote will respond with all objects whose version is later along with a vector of requested objects that are later on the initiating replica. The anti-entropy session is concluded by the initiating peer sending the requested objects.

The speed of total replication, t_r is determined by the size of the system, the rate of anti-entropy, as well as some random element where replicas choose a peer such that no exchange occurs as shown in Equation 1.

$$t_r \approx T \log_3 N + \epsilon \tag{1}$$

Note that this equation gives maximal behavior in the case of accesses happening at every replica, and every replica perfectly selecting a peer. In the case of accesses happening at only one replica, the best case is that all replicas select that peer to perform anti-entropy with.

The system is prone to three types of inconsistencies:

1. *Stale reads*: the replica reads a version that is older than the globally latest version.

2. *Forked writes*: the replica writes to an object such that the parent version is stale.

3. *Stomped writes*: a version becomes outdated so quickly so as to never be fully replicated.

The magnitude of these latencies in a system depends primarily on the latency of connections between replicas (λ_μ) as well as t_r. If $T \gg \lambda_\mu$ then it primarily depends on t_r.

The system is currently implemented as a key-value store called Honu.

Learning Behavior

Adaptivity comes from two learning behaviors: reinforcement learning implemented with multi-armed bandits to optimize the anti-entropy topology and anomaly detection implemented with one class SVMs to detect changes in the client access behavior.

Multi-Armed Bandits

Multi-armed bandits are a class of reinforcement learning algorithm that seek to optimize the search for the probability of a maximal reward. The premise is that you have a set of slot machines (with an arm to pull) and that one of the machines is more likely than the other to produce a jackpot. The goal of the algorithm is to discover which arm is most likely to produce a jackpot.

The algorithm starts by allocating each arm equal probabilities, e.g. a uniform selection of arms. It then selects an arm with the specified probability and observes the result. If the result is a reward, then it increments the probability of that machine being selected, taking away probability density from the others. By continuing with this random selection, eventually the arm with the highest likelihood of payoff is selected.

Most bandit algorithms consider the *exploit vs. explore* tradeoff. E.g. if you discover an arm that's paying off, how often do you continue to use that arm, considering that it could be a local maxima. There are two primary forms of balancing this trade off - epsilon greedy and simulated annealing.

Epsilon greedy algorithms use a parameter, ϵ to modify selection as follows: with a probability, ϵ simply choose the highest likelihood arm (the greedy part), otherwise select from all arms with their specified probabilities. Simulated annealing approaches reduce ϵ over time so that the first phase of the learning is more exploratory and the last phase is more greedy.

To implement this with Anti-Entropy we assign the selection of a remote peer to participate in anti-entropy with a likelihood. The likelihood is modified through reinforcement using various bandit strategies. The reward function can include the following:

1. if data was exchanged between peers

2. the amount of data exchanged between peers

3. the recency of versions in the exchange

4. network latency

5. the locality of accesses to specific objects

A successful anti-entropy session is one where data is exchanged. We are essentially comparing uniform random selection with bandit selection of the remote for an anti-entropy session.

Anomaly Detection

We posit that reinforcement learning produces an optimal topology based on specific client access patterns. However, client access patterns change over time, and the system needs to adapt to these changes as well. Some research will have to be conducted into determining if reinforcement learning strategies are able to adapt themselves. In the meantime, I propose the use of anomaly detection to determine if the normal access pattern of the system has changed.

We will break down the system in to sessions. A session is defined by a period of "normal" accesses, e.g. a specific set of clients accessing objects on specific replicas. During a session, a bandit strategy can be used to optimize the topology of anti-entropy. When the session changes, however, the bandit probabilities can be reset so that a new optimal anti-entropy topology can be learned.

We propose the use of anomaly detection algorithms to detect session boundaries, particularly one-class support vector machines. A time series vector of the number of accesses per client, per object will be created on each replica. The state of the system is all accesses to all replicas across all clients. The SVM will be trained on "stable state" data, e.g. a series of configurations where clients do not shift access patterns very much.

Replicas will also exchange vector information about the state of the system during anti-entropy sessions. If a replica detects an anomaly (e.g. changing state) locally, it will reset its bandit probabilities and also broadcast new session information. Each replica should eventually shift topologies, resulting in a new optimal anti-entropy topology.

Experiments

Related Work

References

[1] Almeida, P. S., Baquero, C., and Fonte, V. Version stamps-decentralized version vectors. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on* (2002), IEEE, pp. 544–551.

[2] Angiulli, F., and Pizzuti, C. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, pp. 15–27.

[3] Bailis, P., Venkataraman, S., Franklin, M. J., Hellerstein, J. M., and Stoica, I. Probabilistically bounded staleness for practical partial quorums. *Proceedings of the VLDB Endowment 5*, 8 (2012), 776–787.

[4] Bouneffouf, D., Laroche, R., Urvoy, T., Fraud, R., and Allesiardo, R. Contextual bandit for active learning: Active thompson sampling. In *International Conference on Neural Information Processing*, Springer, pp. 405–412.

[5] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. Dynamo: amazon's highly available key-value store. *ACM SIGOPS operating systems review 41*, 6 (2007), 205–220.

[6] Haeupler, B. Simple, fast and deterministic gossip and rumor spreading. *Journal of the ACM (JACM) 62*, 6 (2015), 47.

[7] Kalai, A., and Vempala, S. Efficient algorithms for online decision problems. 291–307.

[8] Langford, J., and Zhang, T. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pp. 817–824.

[9] Osugi, T., Kim, D., and Scott, S. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, IEEE, pp. 8–pp.

[10] Parker, D. S., Popek, G. J., Rudisin, G., Stoughton, A., Walker, B. J., Walton, E., Chow, J. M., Edwards, D., Kiser, S., and Kline, C. Detection of mutual inconsistency in distributed systems. *IEEE transactions on Software Engineering*, 3 (1983), 240–247.

[11] Qian, H., Mao, Y., Xiang, W., and Wang, Z. Recognition of human activities using SVM multi-class classifier. 100–111.

[12] Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., and Hauser, C. H. Managing update conflicts in bayou, a weakly connected replicated storage system. In *ACM SIGOPS Operating Systems Review* (1995), vol. 29, ACM, pp. 172–182.

[13] Vogels, W. Eventually consistent. *Communications of the ACM 52*, 1 (2009), 40–44.